How complicated is the set of stable models of a recursive logic program?

W. Marek,[1] A. Nerode[2] and J. Remmel[3]

1 Introduction

1.1 Summary

In Logic Programming, "negation as failure" has become an important area of research. "Negation as failure" is ubiquitous in deductive databases ([Minker, 1987]), in truth maintenance systems ([Doyle, 1979]), and in default logic ([Reiter, 1980]). It is a naturally occurring "non-monotonic logic". The original (breadth-first) PROLOG is a (monotonic) classical logic of pure Horn clauses. There, every Horn program possesses a least Herbrand model. Correspondingly, an important tool for understanding the semantics of PROLOG programs allowing "negation as failure" has been stable models ([Gelfond and Lifschitz, 1988]). But, unlike the situation for pure Horn clause logic, a program here may have no least stable model in the Herbrand universe. In fact, a program may have no stable model or many different stable models. Exactly how complicated can the set of all stable models be? In this paper we determine this for programs with a recursive set of axioms. See [Rogers, 1967] for all unexplained terminology. We show that, up to a 1:1 recursive

[1]Department of Computer Science, University Kentucky, Lexington, KY 40506–0027. Work partially supported by NSF grant RII-8610671 and Kentucky EPSCoR program and ARO contract DAAL03-89-K-0124.

[2]Mathematical Sciences Institute, Cornell University, Ithaca, NY 14853. Work partially supported by NSF grant DMS-8902797 and ARO contract DAAG629-85-C-0018.

[3]Department of Mathematics, University of California at San Diego, La Jolla, CA 92903. Work partially supported by NSF grant DMS-9006413.

renaming, the set of all stable models of a recursive program in the Herbrand universe is the set of all infinite paths in a recursive, countably branching tree (theorem 3.1). One immediate corollary of the relativised version of this result, using Cantor's theorem, says that a countable, not necessarily recursive, logic program has either countably many or a continuum of stable models. Conversely, we show that the set of infinite paths of a countably branching recursive tree is, up to 1:1 recursive renaming, the set of stable models of a recursive program (Theorem 3.3). Thus the problem of finding a stable model of a recursive program and the problem of finding an infinite path through a countably branching recursive tree are essentially equivalent. An important consequence of these correspondences is that the following problem is Σ^1_1 complete: *given a recursive program P, determine whether or not there exists a stable model of P*. That is, using indices of (characteristic functions of) recursive logic programs, we prove that the set of indices of recursive programs possessing stable models is a complete Σ^1_1 set (Corollary 3.4). In addition we show that the set of all indices of recursive programs with at least two stable models is Σ^1_1, that the set of all indices of recursive program with at most one stable model is Π^1_1 (Corollary 3.5), and that the set of all indices of recursive programs with exactly one stable model is Δ^1_2. Moreover our correspondences allow us to apply known results from recursion theory to show the existence of programs whose stable models satisfy any number of special properties. For example, using a result of Hinman ([Hinman, 1978]), we can conclude from our results that for any recursive ordinal α, there exists a recursive program with exactly one stable model, where that model has Turing degree equal to the α^{th} jump of $\mathbf{0}$ (here $\mathbf{0}$ is the degree of the recursive sets).

Our results in this paper stem from the authors' work on non-monotone rule systems ([Marek, Nerode and Remmel, 1990]). The latter is a common framework for a mathematical

development of syntax and semantics and algorithms for many non-monotonic logics which have been offered in the literature, including all those mentioned above. In particular, this work implies that the conclusions here also apply to truth maintenance systems and default logic.

The paper is organized as follows: after the discussion of the problem, in Section 2 we introduce basic technique used throughout the paper. Basic results on correspondence between collections of stable models of a recursive program and effective closed subsets of Baire' space as well as corollaries are proved in Section 3.

1.2 Discussion of the problem

We can rephrase the basic problem that we wish to study as: "What types of objects are described by a logic program?". The first result in this direction is the classical result implicit in ([Smullyan, 1961]) and, more recently, reproved by Andreka and Nemeti ([Andreka and Nemeti, 1978]). This can be summarized as follows: the least model of a recursive Horn program is a recursively enumerable (r.e.) set, and every r.e. set so arises. But general logic programs compute more complex sets. As mentioned above, such a program does not necessarily have a least model. It may possess several or no minimal models. For an important class of general programs, stratified programs and their generalizations, it is possible to single out a distinguished model [Apt, Blair and Walker, 1987], called the perfect model. The perfect model is constructed by combining the least fixed point constructions for a suitably chosen sequence of operators. The arithmetical complexity of perfect models of stratified programs was determined in ([Apt and Blair, 1990]). The results of Apt and Blair showed that programs with stratification of length n compute, up to Turing degree, the n^{th} level of the arithmetical hierarchy.

The concept of a perfect model of a program has been generalized in various ways. When the underlying logic is ordinary two-valued logic, the most natural notion generalizing the concept of a perfect model is that of a stable model of a program [Gelfond and Lifschitz, 1988]. Stable models of programs have been shown to be default interpretations, see [Reiter, 1980], in [Bidoit and Froidevaux, 1988] and in [Marek and Truszczyński, 1989]. There exist logic programs which are not stratified but which possess a unique stable model [Gelfond and Lifschitz, 1988]. Up to now there has been no general characterization of programs which possess a unique stable model. Also, there is no natural method of selecting a particular stable model from the set of stable models of a program with many stable models. Once we accept the supposition that all the stable models of a program are "equally good", that each stable model represents an acceptable "points of view", then it is natural to ask for a characterization of the *class* $\mathcal{S}(P)$ of all stable models of a program P. The authors proved ([Marek, Nerode and Remmel, 1990a]) that for every recursive program P, then the class $\mathcal{S}(P)$ is a Π^0_2 subset of the Cantor space, under a suitable coding of the Herbrand universe as ω. This converts questions about the complexity of stable models to questions in Cantor space or Baire space. So we are presented with a problem in effective descriptive set theory: How can we characterize the effective descriptive complexity of $\mathcal{S}(P)$? Since stable models are minimal models ([Gelfond and Lifschitz, 1988]), the class $\mathcal{S}(P)$ forms an antichain, that is, the elements of $\mathcal{S}(P)$ are pairwise incompatible under inclusion. Our results show that the classes $\mathcal{S}(P)$ are notational variants of the effective closed sets, Π^0_1 sets, in Baire space. What we prove here is that for any given recursive program P, the class $\mathcal{S}(P)$ is in effective one-to-one degree preserving correspondence with a Π^0_1 subset of Baire space. Conversly, every Π^0_1 set so arises.

2 Operator $T_{P,M}$. Parametrized derivations. Derivation schemes for logic programs.

Let Π_P be the set of all the Herbrand (ground) substitutions of the logic program P. We identify P with Π_P for the rest of the paper. If P is a program and $M \subseteq B_P$ is a subset of the Herbrand base, define operator $T_{P,M}: \mathcal{P}(B_P) \rightarrow \mathcal{P}(B_P)$ as follows:

$$T_{P,M}(I) = \{p: \text{there exist a clause } C = p \leftarrow q_1, \ldots, q_r, \neg s_1, \ldots, \neg s_u$$

$$\text{in } P \text{ such that } q_1 \in I, \ldots, q_r \in I, s_1 \notin M, \ldots s_u \notin M\}\}$$

The following is immediate:

Proposition 2.1 *For every program P and every subset M of B_P, the operator $T_{P,M}$ is monotone and finitizable.*

(See [Apt, 1988] for unexplained notions.)

The operator $T_{P,M}$, like all monotonic finitizable operators, possesses a least fixpoint $F_{P,M}$. In order to understand this fixpoint in the context of the models of the program P, recall the operational construction of a stable model of a logic program from ([Gelfond and Lifschitz, 1988]).

Given program P and $M \subseteq B_P$, first define the *Gelfond-Lifschitz reduct* of P as follows. For every clause C of P, execute the following operation: If some atom a belongs to M and its negation $\neg a$ appears in C, then eliminate C altogether. In the remaining clauses that have not been eliminated by the operation above, eliminate all the negated atoms.

The resulting program P_M^{GL} is a Horn propositional program (possibly infinite). The program P_M^{GL} possesses a least Herbrand model. If that least model of P_M^{GL} coincides with

M, then M is called a *stable structure* for P. Gelfond and Lifschitz proved the following result:

Proposition 2.2 ([Gelfond and Lifschitz, 1988]) *If M is a stable structure for P, then M is a minimal model of P.*

Call a stable structure for P a *stable model* of P. Moreover, as shown in [Marek and Subrahmanian, 1988], stable models of P are all minimal models of the completion of P ([Clark, 1978]). The Gelfond and Lifschitz construction is an operational reduction of the construction of stable models to monotone, finitizable operators. This reduces the study of the stability phenomenon to the study of the usual operator T_P of van Emden and Kowalski [1976].

One condition which ensures that P possesses a unique stable model is that P is stratified ([Gelfond and Lifschitz, 1988]). A similar result is proved in [Marek and Subrahmanian, 1988] for locally stratified programs. But in general P may possess infinitely many, and even uncountably many, stable models.

We start from the following natural characterization of stability in terms of the fixpoints of the operator $T_{P,M}$.

Proposition 2.3 *M is a stable model for program P if and only if $M = F_{P,M}$.*

Proof: Assume that M is a stable model of P. Let $T^{GL}_{P,M}$ be the operator associated with the Horn program P^{GL}_M. We prove by induction on n that for every $n < \omega$, $T_{P,M\Uparrow n}(\emptyset) = T^{GL}_{P,M\Uparrow n}(\emptyset)$. For $n = 0$, the elements of $T_{P,M\Uparrow 0}(\emptyset)$ are precisely those atoms p for which "$p \leftarrow$" belongs to P^{GL}_M. Such a "$p \leftarrow$" is either directly an element of P or, for some atoms $q_1, \ldots, q_u \notin M$, $p \leftarrow \neg q_1, \ldots, \neg q_u \in P$. In both cases $p \in T_{P,M\Uparrow 0}(\emptyset)$. Conversely,

if $p \in T_{P,M\Uparrow 0}(\emptyset)$, then there is a $C \in P$ such that $C = p \leftarrow \neg q_1, \ldots, \neg q_u \in P$ and $q_1, \ldots, q_u \notin M$. Then $p \leftarrow \in P^{GL}_M$ and $p \in T^{GL}_{P,M\Uparrow n}(\emptyset)$.

The inductive step is similar and is left to the reader.

Consequently, $M = \bigcup_{n<\omega} T_{P,M\Uparrow n}(\emptyset)$ if and only if $M = \bigcup_{n<\omega} T^{GL}_{P,M\Uparrow n}(\emptyset)$. That is, $M = \bigcup_{n<\omega} T^{GL}_{P,M\Uparrow n}(\emptyset)$ if and only if $M = F_{P,M}$. □

Having characterized stable models as fixpoints of (parametrized) operators, we look at the form of elements belonging to $F_{P,M}$.

A P, M-*derivation* of an atom p is a sequence $\langle p_1, \ldots, p_s \rangle$ such that:

(1) $p_s = p$,

(2) for every $i \leq s$, either "$p_i \leftarrow$" is a member of P or there is a clause $C =$ "$p_i \leftarrow r_1, \ldots, r_n, \neg q_1, \ldots, \neg q_m$" such that $C \in P$, $r_1, \ldots, r_n \in \{p_1, \ldots, p_{i-1}\}$, and $q_1, \ldots, q_m \notin M$.

Proposition 2.4 *$F_{P,M}$ is the set of all atoms possessing a P, M-derivation.*

Proof: Both inclusions may be proved by a double induction on the number of iterations of $T_{P,M}$ and on the length of the P, M-derivation. □

Proposition 2.3 says that M is a stable model of the program P if and only if M consists exactly of those atoms which possess a P, M-derivation. This fixpoint characterization of stability explains the existence of programs with multiple stable models, and also the existence of programs without any stable model.

The property that a sequence $\langle p_1, \ldots, p_s \rangle$ is a P, M-derivation of an atom p does not depend on the whole set M but only on the behavior of M on a finite set of atoms. In order that the sequence $\langle p_1, \ldots, p_s \rangle$ be a P, M-derivation of an atom p_s, some atoms must be *left out of* the set M. Each derivation depends on a *finite* number of such omitted atoms. In

other words, if we classify the atoms according to whether they are "in" or "out" of M, the property that a sequence $\langle p_1, \ldots, p_s \rangle$ is a P, M-derivation depends only on whether a finite number of elements are out of M. We formalize this notion.

An (*annotated*) *(P-)proof scheme* for an atom p is a sequence $S = \langle \langle p_i, C_i, u_i \rangle \rangle_{i=1}^{s}$ of triples such that for each triple $\langle p_i, C_i, u_i \rangle$, $p_i \in B_P$, $C_i \in P$ is a clause with the head p_i and u_i is a finite subset of B_P. Such sequence S is a *proof scheme* for p if:

(1) $p_s = p$,

and for every i

(2) $C_i = p_i \leftarrow r_1, \ldots, r_n, \neg q_1, \ldots, \neg q_m$, where $\{r_1, \ldots, r_n\} \subseteq \{p_1, \ldots p_{i-1}\}$ and $u_i = u_{i-1} \cup \{q_1, \ldots, q_m\}$.

(3) $\{p_1, \ldots, p_s\} \cap u_s = \emptyset$.

We call p the *conclusion* of S and denote this by writing $p = cln(S)$. Call the set u_m the *support* of S.

The notion of a proof scheme is needed because $\langle p_1, \ldots, p_s \rangle$ may be a P, M-derivation for many different M's. All that is needed for the sequence $\langle p_1, \ldots, p_s \rangle$ to be a P, M-derivation is that M satisfies certain negative requirements. These requirements are accumulated by condition (2) above. The idea behind condition (3) is that if M is a stable model and the derivation $\langle p_1, \ldots, p_s \rangle$ requires that some atoms be out of M, but at the same time proves one of these atoms, then this derivation can never be used to establish that its conclusion is in M.

One needs to realize that, whereas the notion of P, M-derivation has been defined for one model M, in contrast, the notion of P-proof scheme does not depend on M. We say that a subset $M \subseteq B_P$ *admits* a proof scheme $S = \langle \langle p_i, C_i, u_i \rangle \rangle_{i=1}^{s}$ if $M \cap u_s = \emptyset$. We have

the following fact.

Proposition 2.5 *If M admits a proof scheme $S = \langle\langle p_i, C_i, u_i\rangle\rangle_{i=1}^{s}$, then $\langle p_1, \ldots, p_s\rangle$ is a P, M-derivation of p_s.*

Proof: The result follows by a straightforward induction on s. □

Thus, Proposition 2.5 characterizes of stability in terms of the existence of proof schemes.

Proposition 2.6 *Let $M \subseteq B_P$. Then M is a stable model of P if and only if*

(1) for every $p \in M$, there is a proof scheme S for p such that M admits S, and

(2) for every $p \notin M$, there is no proof scheme S for p such that M admits S.

Proof: If M is a stable model for P and $p \in M$, then there exists a P, M-derivation $\langle p_1, \ldots, p_s\rangle$ of p. Since M is stable, for all $i \leq s$, the atom p_i belongs to M. Let $C_1, \ldots, C_s$ be the clauses used in the derivation. Let v_i be the collection of atoms appearing negatively in C_i. Then $v_i \cap M = \emptyset$. Hence $\{p_1, \ldots, p_s\} \cap v_i = \emptyset$. Thus if we set $u_i = \bigcup_{j \leq i} v_i$, we get that $S = \langle\langle p_i, C_i, u_i\rangle\rangle_{i=1}^{s}$ is a proof scheme for p admitted by M.

If $p \notin M$ then, by proposition 2.5 there is no proof scheme for p admitted by M.

Conversely, assume that (1) and (2) hold for a particular M. Then all the elements of M possess a P, M-derivation, but no other element does. Then by Proposition 2.3, M is a stable model of P. □

But how many derivation schemes for an atom p can there be? If we allow P to be infinite, then it is easy to construct an example with infinitely many derivations of a single atom. Moreover given two proof schemes, one can insert one into the other (increasing appropriately the sets u_i in this process, with obvious restrictions). Thus various clauses

C_i may be immaterial to the purpose of deriving p. This leads us to introduce a natural relation $\prec$ on proof schemes using a well-known device from proof theory. Namely, we define $S_1 \prec S_2$ if S_1, S_2 have the same conclusion and if every clause appearing in S_1 also appears in S_2. Then a *minimal* proof scheme for p is defined to be a proof scheme S for p such that whenever S' is a proof scheme for p and $S' \prec S$, then $S \prec S'$. Note that $\prec$ is reflexive and transitive, but $\prec$ is not antisymmetric. However it is wellfounded. That is, given any proof scheme S , there is an S' such that $S' \prec S$ and for every S'', if $S'' \prec S'$ then $S' \prec S''$. Moreover, the associated equivalence relation, $S \equiv S'$, defined by $S \prec S'$ and $S' \prec S$, has finite equivalence classes.

Example 2.1 *Let P_1 be the following program:*

C_1: $p(0) \leftarrow \neg q(Y)$.

C_2: $nat(0) \leftarrow$.

C_3: $nat(s(X)) \leftarrow nat(X)$.

Then atom $p(0)$ possesses infinitely many minimal proof schemes. For instance, each one-element sequence:

$$S_i = \langle\langle p(0), C_1\Theta_i, \{s_i(0)\}\rangle\rangle$$

where Θ_i is the operation of substituting $s^i(0)$ for Y, is a minimal proof scheme for $p(0)$.

Example 2.2 *Let P_2 be the following program:*

C_1: $q(s(Y)) \leftarrow \neg q(Y)$.

C_2: $nat(0) \leftarrow$.

C_3: $nat(s(X)) \leftarrow nat(X)$.

For P_2, each atom possesses only finitely many minimal proof schemes.

We shall call a program P *locally finite* if for every atom p, there are only finitely many *minimal* proof schemes with conclusion p. The assumption of local finiteness implies that for every p, there is a *finite* subset $D_p \subseteq B_P$ such that, for every M, the behavior of M *on* D_p (that is, the partition $D_p = (D_p \cap M) \cup (D_p \setminus M)$) determines whether or not p possesses a P, M-derivation. This D_p is the union of the supports of the minimal proof schemes for p. This implies that if P is locally finite, when we attempt to construct a subset $M \subseteq B_P$ which is a stable structure for P, we can apply a straightforward (although still infinite) tree construction to produce such an M, if such an M exists at all.

Next, we need to make the notion of a recursive program precise. First, assume that we have a Gödel numbering of the elements of the Herbrand base B_P. Thus, we can think of each element of the Herbrand base as a natural number. If $p \in B_P$, write $c(p)$ for the code or Gödel number of p. Let $\omega = \{0, 1, 2, \ldots\}$. Assume $[,]$ is a fixed recursive pairing function $[,]$ which maps $\omega \times \omega$ onto ω and has recursive projection functions π_1 and π_2, defined by $\pi_i([x_1, x_2]) = x_i$ for all x_1 and x_2 and $i \in \{0, 1\}$. Code a finite sequence $\langle x_1, \ldots, x_n \rangle$ for $n \geq 3$ by the usual inductive defintion $[x_1, \ldots, x_n] = [x_1, [x_2, \ldots, x_n]]$. Next, code finite subsets of ω via "canonical indices". The *canonical index* of the empty set, $\emptyset$, is the number 0 and the set $\{x_0, \ldots, x_n\}$, where $x_0 < \ldots < x_n$, has canonical index $\sum_{j=0}^{n} 2^{x_j}$. Let E_k denote the finite set whose canonical index is k. Once finite sets and sequences of natural numbers have been coded, we can code more complex objects such as clauses, proof schemes, etc. as follows. Let the code $c(C)$ of a clause $C = p \leftarrow r_1, \ldots, r_n, \neg q_1, \ldots, \neg q_m$ be $[c(p), k, l]$, where k is the canonical index of the finite set $\{c(r_1), \ldots, c(r_n)\}$, and l is the canonical index of the finite set $\{c(q_1), \ldots, c(q_m)\}$. Similarly, let the code $c(S)$ of a proof scheme $S = \langle \langle p_i, C_i, u_i \rangle \rangle_{i=1}^{s}$ be $[s, [[c(p_1), c(C_1), c(u_1)], \ldots, [c(p_s), c(C_s), c(u_s)]]]$, where for each i, $c(u_i)$ is the canonical index of the finite set of codes of the elements of u_i. The first

coordinate of the code of a proof scheme is the length of the proof scheme. Once we have defined the codes of proof schemes then for locally finite programs we can define the code of the set D_p consisting of the union of the supports of all minimal proof schemes for P. Finally we code recursive sets as natural numbers. Let $\phi_0, \phi_1, \ldots$ be an effective list of all partial recursive functions then ϕ_e is the partial recursive function computed by the e-th Turing machine. By definition, a (recursive) index of a recursive set R is an e such that ϕ_e is the characteristic function of R. Call a program P *recursive* if the set of codes of the Herbrand universe B_P is recursive and the set of codes of the clauses of the program P is recursive. If P is a recursive program, then by an index of P we mean the code of a pair $[u, p]$ where u is an index of the recusive set of all codes of elements in B_P and p is an index of the recursive set of the codes of all clauses in P.

For the rest of this paper we shall identify an object with its code as described above. This means that we shall think of the Herbrand universe of a program, and the program itself, as subsets of ω and clauses, proof schemes, etc. as elements of ω.

Even if P is a locally finite progam, there is no guarantee that the global behaviour of the function $p \mapsto D_p$, mapping ω into ω, has any sort of effective properties. Thus we are led to define the following.

We say that a locally finite recursive program P possesses a *recursive proof structure* (rps) if:

(1) P is locally finite, and

(2) The function $p \mapsto D_p$ is recursive.

A locally finite recursive program with an rps is called an *rps program.*

In a forthcoming paper [Marek, Nerode and Remmel, 1991], we characterize the complexity of the set of stable models for locally finite recursive programs and also the set of stable models of a locally finite recursive program with a recursive proof structure. Here we concentrate only on the most general case: recursive programs with no restrictions on the number of minimal derivations.

3 Paths through Binary Trees and Stable Models

In this section, we show that the problem of finding a stable model for a recursive program and the problem of finding a path through a recursive tree in $\omega^{<\omega}$ are recursively equivalent. To make this statement precise, we need yet more notation. Recall that $[,]:\omega \times \omega \to \omega$ is a fixed one-to-one and onto recursive pairing function such that the projection functions π_1 and π_2 defined by $\pi_1([x,y]) = x$ and $\pi_2([x,y]) = y$ are also recursive. Let $\omega^{<\omega}$ denote the set of all finite sequences from ω and let $2^{<\omega}$ denote the set of all finite sequences of 0's and 1's. Given $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ and $\beta = \langle \beta_1, \ldots, \beta_k \rangle$ in $\omega^{<\omega}$, write $\alpha \sqsubseteq \beta$ if α is initial segment of β; that is, if $n \leq k$ and $\alpha_i = \beta_i$ for $i \leq n$. For the rest of this paper, identify a finite sequence $\alpha = \langle \alpha_1, \ldots, \alpha_n \rangle$ with its code $c(\alpha) = [n, [\alpha_1, \ldots, \alpha_n]]$ in ω. Let 0 be the code of the empty sequence $\emptyset$. Thus, when we say that a set $S \subseteq \omega^{<\omega}$ is recursive (recursively enumerable, etc.), we will mean that the set $\{c(\alpha): \alpha \in S\}$ is recursive, (recursively enumerable, etc.) A *tree* T is a nonempty subset of $\omega^{<\omega}$ closed under initial segments. A tree T contained in $\omega^{<\omega}$ is *recursive* if the set of codes of nodes in T is a recursive subset of ω. By a (recursive) index of a tree $T \subseteq \omega^{<\omega}$, we mean an index of the characteristic function of the recursive set consisting of all codes of nodes in T. We shall identify a tree T contained in $\omega^{<\omega}$ with the set of codes of the nodes in T. Thus we think of T as a certain subset of ω. Suppose that T is a tree contained in $\omega^{<\omega}$, then a function $f:\omega \to \omega$ is called an infinite *path*

through T if for all n, $< f(0), \ldots, f(n) > \in T$. Let $[T]$ denote the set of all infinite paths through T. A set A of functions is called a Π^0_1-class if there is a recursive predicate R such that $A = \{f: \omega \to \omega : \forall_n(R([f(0), \ldots, f(n)])\}$. Note that if T is a tree contained in $2^{<\omega}$, then $[T]$ is a collection of $\{0,1\}$-valued functions. Identify each $f \in [T]$ with the set A_f, $A_f = \{x: f(x) = 1\}$ and think of $[T]$ as a Π^0_1 class of sets.

We say that there is an effective one-to-one degree preserving correspondence between the set of stable models of a recursive program P, $\mathcal{S}(P)$, and the set of infinite paths $[T]$ through a recursive tree T if there are indices e_1 and e_2 of oracle Turing machines such that

(i) $\forall_{f \in [T]}\{e_1\}^{gr(f)} = M_f \in \mathcal{S}(P)$,

(ii) $\forall_{M \in \mathcal{S}(P)}\{e_2\}^M = f_M \in [T]$, and

(iii) $\forall_{f \in [T]}\forall_{M \in \mathcal{S}(P)}(\{e_1\}^{gr(f)} = M$ if and only if $\{e_2\}^M = f)$.

Here $\{e\}^B$ denotes the function computed by the e^{th} oracle machine with oracle B. We write $\{e\}^B = A$ for a set A if $\{e\}^B$ is a characteristic function of A. If f is a function $f: \omega \to \omega$, then $gr(f) = \{[x, f(x)]: x \in \omega\}$. Condition (i) says that the branches, that is the infinite paths of the tree T, uniformly produce stable models via an algorithm with index e_1. Condition (ii) says that stable models of P of $\mathcal{S}$ uniformly produce branches of the tree T via an algorithm with index e_2. A is *Turing reducible* to B, written $A \leq_T B$, if $\{e\}^A = B$ for some e. A is *Turing equivalent* to B, written $A \equiv_T B$, if both $A \leq_T B$ and $B \leq_T A$. Thus condition (iii) asserts that our correspondence is one-to-one and if $\{e_1\}^{gr(f)} = M_f$, then f is Turing equivalent to M_f. In what follows we will not explicitly construct indices e_1 and e_2, but it will be clear that such indices exist in each case.

We are ready to establish the relationship between stable models of recursive programs and effective closed (Π^0_1) subsets of Baire space. For any unexplained notion of effective descriptive set theory see [Rogers, 1967].

Theorem 3.1 *Let P be a recursive program. Then there exists a recursive tree $T \subseteq \omega^{<\omega}$ and an effective one-to-one degree preserving correspondence between the set $\mathcal{S}(P)$ of stable models of the program P, and the set $[T]$ of all infinite branches of the tree T.*

Proof: Without loss of generality, we can assume that the Herbrand base of the program P is ω. This can be always assured by adding superfluous atoms. This preserves the concept of stable model. Thus we assume that $B = \omega$.

Since the program P is recursive, the family $\mathcal{M}(P)$ of codes for minimal proof schemes relative to P is also recursive. In fact, for every atom p, the set $\mathcal{M}(P, p)$ of codes for minimal proof schemes d with $cln(d) = p$ is also recursive.

We code a stable model M of program P by a path $P_M = \langle \pi_0, \pi_1, \pi_2 \ldots \rangle$ through the complete ω-branching tree $T_\omega = \omega^{<\omega}$ as follows. First, for all $i > 0$, $\pi_{2i} = \chi_M(i)$. Next, if $\pi_{2i} = 0$, then also $\pi_{2i+1} = 0$. If, however, $\pi_{2i} = 1$, so that $i \in M$, then $\pi_{2i+1} = d_M(i)$ where $d_M(i)$ is the least d such that d is the code of a minimal proof scheme $P = \langle \langle p_0, C_0, G_0 \rangle, \ldots, \langle p_m, C_m, G_m \rangle \rangle$ and $p_m = i$, and $G_m \subseteq \omega \setminus M$. Since M is a stable model for P, such a proof scheme must exist. Clearly $M \leq_T \pi_M$. Given an oracle for M, it is easy to see that for each $i \in M$, we can use the oracle to effectively find $d_M(i)$. Therefore $\pi_M \leq_T M$. It thus follows that the correspondence $M \mapsto \pi_M$ is an effective one-to-one degree preserving correspondence. All that is left is to prune the full tree T_ω to a tree $T \subseteq \omega^{<\omega}$ such that $[T] = \{\pi_M : M \in \mathcal{S}(P)\}$.

Let N_k be the set of all codes of minimal proof schemes $D = \langle \langle p_0, C_0, G_0 \rangle, \ldots, \langle p_m, C_m, G_m \rangle \rangle$ such that every atom appearing in any C_i for $i \leq m$ is contained in $\{0, \ldots, k\}$. Note that N_k is finite. Moreover, a canonical index of N_k can be uniformly computed from k. Given a node $\sigma = \langle \sigma(0), \ldots, \sigma(k) \rangle \in \omega^{<\omega}$, let $I_\sigma = \{i : 2i \leq k \wedge \sigma(2i) = 1\}$. Let

$O_\sigma = \{i: 2i \leq k \wedge \sigma(2i) = 0\}$. Define T by putting $\sigma = \langle \sigma(0), \ldots, \sigma(k)\rangle$ in T if and only if:

(a) $\forall_i(2i \leq k \wedge \sigma(2i) = 0 \wedge 2i+1 \leq k \Rightarrow \sigma(2i+1) = 0)$.

(b) $\forall_i(2i \leq k \wedge \sigma(2i) = 1 \wedge 2i+1 \leq k \Rightarrow \sigma(2i+1) = d$, where the number d is a code of a minimal proof scheme $D = \langle\langle p_0, C_0, G_0\rangle, \ldots \langle p_m, C_m, G_m\rangle\rangle$ such that $p_m = i$ and $G_m \cap I_\sigma = \emptyset$).

(c) $\forall_i(2i \leq k \wedge \sigma(2i) = 1 \wedge 2i+1 \leq k \Rightarrow$ there is no code $c \in N_{\lfloor k/2 \rfloor}$ of a minimal proof scheme $\overline{D} = \langle\langle \overline{p}_0, \overline{C}_0, \overline{G}_0\rangle, \ldots \langle \overline{p}_m, \overline{C}_m, \overline{G}_m\rangle\rangle$ such that $\overline{p}_m = i$, $\overline{G}_m \subseteq O_\sigma$, and $c < \sigma(2i+1))$.

(d) $\forall_i(2i \leq k \wedge \sigma(2i) = 0 \Rightarrow$ there is no code $c \in N_{\lfloor k/2 \rfloor}$ of a minimal proof scheme $\overline{\overline{D}} = \langle\langle \overline{\overline{p}}_0, \overline{\overline{C}}_0, \overline{\overline{G}}_0\rangle, \ldots \langle \overline{\overline{p}}_m, \overline{\overline{C}}_m, \overline{\overline{G}}_m\rangle\rangle$ such that $\overline{\overline{p}}_m = i$, $\overline{\overline{G}}_m \subseteq O_\sigma)$.

(Here $\lfloor t \rfloor$ denotes the greatest integer $\leq t$).

It is quite easy to see that T is a tree. That is, if $\sigma \in T$ and $\tau \sqsubseteq \sigma$, then $\tau \in T$. Moreover it should be clear from our definition of T that T is a recursive subset of $\omega^{<\omega}$. Thus T is a recursive tree.

In case M is a stable model of P and $\pi_M = \langle \pi_0, \pi_1, \ldots\rangle$, it is routine to check that $\langle \pi_0, \ldots, \pi_n\rangle \in T$ for all n. Thus $\pi_M \in [T]$. Conversely, assume that $\beta = \langle \beta_0, \beta_1 \ldots\rangle \in [T]$. Define $M_\beta = \{\beta_{2i} = 1\}$. We must show that M_β is a stable model for P. Suppose not. Then:

(a) there exists an i belonging to $M_\beta \setminus F_{M_\beta, P}$, or

(b) there exists an i belonging to $F_{M_\beta, P} \setminus M_\beta$.

We show that neither (a) nor (b) is possible.

Suppose (a) held. Then consider β_{2i+1}. For $\langle \beta_0, \ldots, \beta_{2i+1}\rangle = \beta^{(2i+1)}$ to be in T, it must

be that β_{2i+1} is a code of a minimal proof scheme $\langle\langle p_0, C_0, G_0\rangle, \ldots \langle p_m, C_m, G_m\rangle\rangle$ such that $p_m = i$, and $G_m \cap I_{\beta^{(2i+1)}} = \emptyset$. However, since $i \notin F_{M_\beta,P}$, there must be some $n \in G_m \cap M_\beta$. But then $\beta^{(2n)} = \langle\beta_0, \ldots, \beta_{2n}\rangle \notin T$ because $G_m \cap I_{\beta^{(2n)}} \neq \emptyset$. Thus there is a k such that $\beta^k \notin T$. So $\beta \notin [T]$.

Suppose (b) held. Then for some n, there is a minimal proof scheme $\overline{D} = \langle\langle\overline{p}_0, \overline{C}_0, \overline{G}_0\rangle, \ldots,$ $\langle\overline{p}_m, \overline{C}_m, \overline{G}_m\rangle\rangle$ such that $\overline{p}_m = j$, and $\overline{G}_m \subseteq O_{\beta^n}$. But then $\beta^{(n)} = \langle\beta_0, \ldots, \beta_n\rangle$ does not satisfy condition (d) of our definition for $\beta^{(n)}$ to be in T. Thus $\beta^{(n)} \notin T$, and hence $\beta \notin [T]$. Thus if $\beta \in [T]$, it must be the case that M_β is a stable model of P. Finally, we claim that if $\beta \in [T]$ then $\beta = \pi_{M_\beta}$. Indeed, if $\beta \neq \pi_{M_\beta}$ then for some $i \in M_\beta$, there is a code c of a minimal proof scheme $\langle\langle p_0, C_0, G_0\rangle, \ldots \langle p_m, C_m, G_m\rangle\rangle$ such that $p_m = i$, $G_m \subseteq \omega \setminus M_\beta$, and $c < \beta_{2i+1}$. But then there is an n large enough so that $G_m \subseteq O_{\beta^{(n)}}$. Hence $\beta^{(n)} = \langle\beta_0, \ldots, \beta_n\rangle$ does not satisfy condition (c) of our definition for $\beta^{(n)}$ to be in T. Hence, if $\beta \neq \pi_{M_\beta}$, then $\beta^{(n)} \notin T$ for some n. So $\beta \notin [T]$. We have proved that $\beta \in [T]$ if and only if M_β is a stable model for P and $\beta = \pi_{M_\beta}$. □

The proof of Theorem 3.1 is uniform in indices of recursive programs. That is, the proof the Theorem 3.1 gives an algorithm which, from an index of a recursive program P, produces an index of a recursive tree T and an effective one-to-one degree preserving correspondence between the set of stable models of P and the set of infinite paths through T. It follows that there is a recursive one-to-one function h with the property that if e is the index of a recursive program P, then $h(e)$ is the index of a recursive tree T with an associated one-to-one effective degree preserving correspondence between $\mathcal{S}(P)$ and $[T]$.

One can easily modify the basic algorithm implicit in the proof of Theorem 3.1 to ensure that if e is not an index of a recursive program P, then the algorithm produces an index of a partial recursive function $\phi_{f(e)}$ such that $\phi_{f(e)}$ is not the characteristic function

of a recursive tree contained in $\omega^{<\omega}$. That is, given e, we first find k and l such that $e =< k, l >$. We must check that ϕ_k and ϕ_l are total recursive functions whose range is contained in $\{0, 1\}$, that for each x such that $\phi_l(x) = 1$, x is a code of a clause involving atoms from the set $U = \{z : \phi_k(z) = 1\}$, and that U is a tree contained in $\omega^{<\omega}$. Then the basic idea is to start to compute $\phi_k(0), \phi_l(0), \phi_k(1), \phi_l(1), \ldots$ via the usual dovetailing of computations (compute one step in the computations of $\phi_k(0)$ and $\phi_l(0)$, then compute two steps in the computations of $\phi_k(0), \phi_l(0), \phi_k(1)$ and $\phi_l(1)$, etc). Whenever we are successful in computing $\phi_l(x) = 1$, we then compute a,b, and c such that $x =< a, b, c >$. Now x is supposed to be a code of a clause, so the elements in $B_x = \{a\} \cup E_b \cup E_c$ must be elements of the Herbrand base. So we compute $\phi_k(y)$ and verify that $\phi_k(y) = 1$ for all $y \in B_x$. Similarly, if $\phi_k(z) = 1$, we must find a sequence $\langle \sigma_0, \ldots, \sigma_s \rangle$ whose code is z and check that the codes of $\langle \sigma_0, \ldots, \sigma_r \rangle$ for $r \leq s$ are in U. As we carry out these computations, we alternately also carry our the computations to construct our tree, assuming that e is the index of a program. (Of course, to define the tree, we will need to carry out various computations about the program.) Our algorithm will thus ensure that no step in constructing the tree can be performed unless all the information about the program that is required to perform such a step has been computed by our dovetailing procedure. There may be some problems with e being the index of tree; for example if either ϕ_k or ϕ_l is not total or we find that there is an x for which $\phi_l(x) = 1$ but it is not the case that $\phi_k(y) = 1$ for all $y \in B_x$, etc. Then we will not give a total description of the tree and we will end up decribing only the index of a partial recursive function. In this way we can show that there exists a recursive one-to-one function f such that $f(e)$ is an index of a recursive tree $T \subseteq \omega^{<\omega}$ if and only if e is an index of a recursive program P. Moreover, if e is an index of a recursive program P, then $f(e)$ is an index of a recursive tree T such that there is an effective one-to-one degree

preserving correspondence between $\mathcal{S}(P)$ and $[T]$.

If A and B are subsets of ω, then A is called *1-reducible* to B (written $A \leq_1 B$), if there is a one-to-one recursive function g such that for all $x \in \omega$, $x \in A$ if and only if $g(x) \in B$. Also A is said to be *1-equivalent* to B (written $A \equiv_1 B$), if $A \leq_1 B$ and $B \leq_1 A$. Thus the existence of the one-to-one recursive function f described above shows that *Stab* $= \{e: e$ is the index of a recursive program P such that P has a stable model$\}$ is 1-reducible to *Infpath* $= \{r: r$ is the index of a recursive tree T such that T has an infinite path$\}$ and that *Nostab* $= \{e: e$ is the index of a recursive program P without stable model $\}$ is 1-reducible to *Finpath* $= \{r: r$ is the index of a recursive tree T such that T has no infinite path$\}$.

Theorem 3.1 relativizes. That is, even if P is not recursive, the resulting tree is recursive in P. Therefore the collection of stable models of P is in a one-to-one correspondence with a closed subset of Baire space. Using Cantor's theorem on the cardinality of closed subsets we get the following

Corollary 3.2 *If P is a logic program, then either P has at most denumerable number of stable models, or P has exactly $2^{\aleph_0}$ stable models.*

Next, we prove a result which is the converse of the Theorem 3.1. That is, we encode an effective closed subset of Baire space as the set of of stable models of a recursive program.

Theorem 3.3 *Let T be a recursive tree contained in $\omega^{<\omega}$. Then there exists a recursive program P with an associated effective one-to-one degree preserving correspondence between the collection $[T]$ of all paths through T and the collection $\mathcal{S}(P)$ of all stable models of P.*

Proof: Consider the collection of atoms B consisting of the union of three sets: $\{p_\sigma: \sigma \in T\}$, $\{p_{\overline{\sigma}}: \sigma \in T\}$, and $\{p_0, p_1, \ldots\}$. We identify each atom p in this union with its code $c(p)$,

where $c(p)$ is defined as follows:

(a) if $\sigma = \langle \sigma(0), \ldots, \sigma(k) \rangle$ then $c(p_\sigma) = 2[0, k, \sigma(0), \ldots, \sigma(k)]$.

(b) if $\sigma = \langle \sigma(0), \ldots, \sigma(k) \rangle$ then $c(p_{\overline{\sigma}}) = 2[1, k, \sigma(0), \ldots, \sigma(k)]$.

(c) $c(p_\emptyset) = 0$, $c(p_{\overline{\emptyset}}) = 1$ and

(d) $c(p_i) = 2i + 3$ for $i \geq 0$.

For any node $\sigma = \langle \sigma(0), \ldots, \sigma(k) \rangle$, let $\sigma \circ n$ denote the node $\langle \sigma(0), \ldots, \sigma(k), n \rangle$. Our program P consists of seven classes of clauses.

1. $p_{\overline{\sigma}} \leftarrow \neg p_\sigma$

   $p_\sigma \leftarrow \neg p_{\overline{\sigma}}$

2. $p \leftarrow p_\sigma, p_{\overline{\sigma}}$, for all $\sigma \in T$, $p \in B$.

3. $p_{\overline{\sigma \circ j}} \leftarrow p_{\sigma \circ n}$, for all σ such that $\sigma \circ n, \sigma \circ j \in T$, $j \neq n$.

4. $p_{\overline{\sigma \circ n}} \leftarrow p_{\overline{\sigma}}$, whenever $\sigma \in T$ and $\sigma \circ n \in T$.

5. $p_k \leftarrow p_\sigma$ whenever $\sigma \in T$, $| \sigma | = k$.

6. $p \leftarrow \neg p_k$ for all $k \geq 0$, $p \in B$.

7. $p_\emptyset \leftarrow$.

Then P is a recursive program, since T is recursive. Now suppose that $\pi : \omega \rightarrow \omega$ is an infinite path through T. Define:

$$M_\pi = \{p_i : i \in \omega\} \cup \{p_{\pi|_n} : n \in \omega\} \cup \{p_\emptyset\} \cup \{p_{\overline{\sigma}} : \sigma \notin \{\emptyset\} \cup \{\pi |_n : n \in \omega\}\},$$

where $\pi |_n = \langle \pi_0, \ldots, \pi_n \rangle$.

We claim that $\mathcal{S}(P)$ is precisely $\{M_\pi : \pi \in [T]\}$. It is easy to see that the map $\pi \mapsto M_\pi$ is an effective one-to-one degree preserving correspondence, so that P will be the desired recursive program once we prove our claim.

The first step in proving our claim is to show that that B is not a stable model. To this end, notice that only clauses of the form (1), (6), and (7) have no premises. The application of any clause in (1) or (6) is blocked by B. Clause (7) implies that $p_\emptyset \in F_{M,P}$. Then, using clause (5), we derive that $p_0 \in F_{P,M}$. In fact, $F_{P,M} = \{p_\emptyset, p_0\}$. This can easily be proved by induction on the lengths of proof schemes. Thus $T_{P,B\Uparrow\omega} \neq B$, and so B is not a stable model of P.

Suppose M were a stable model of P. Clauses of the form (2) or (6) can never be used in the M-derivations since $M \neq B$. By (6), we can conclude that $p_k \in M$ for all k. Since clauses of type (6) cannot be used in M-derivation and we already established that p_k is in M, p_k must be derived in some other fashion. It is easy to see that the only way p_k can be derived is by an application of a clause of type (5). This implies that for every $k \in \omega$, there is a $\sigma \in T$ such that $\mid \sigma \mid = k$ and $p_\sigma \in M$. Next, the clauses of type (1) guarantee that at least one of $p_\sigma, p_{\overline{\sigma}}$ belongs to M. That is, it can not be the case that both p_σ and $p_{\overline{\sigma}}$ are not in M, since otherwise clauses of type (1) would show that both p_σ and $p_{\overline{\sigma}}$ are in $T_{P,M\Uparrow\omega}$, and hence that $T_{P,M\Uparrow\omega} \neq M$, violating our assumption that M a stable model of P. On the other hand, since $M \neq B$, at most one of $p_\sigma, p_{\overline{\sigma}}$ belongs to M due to clauses of type (2). Therefore, for every $\sigma \in T$, $\mid \{p_\sigma, p_{\overline{\sigma}}\} \cap M \mid = 1$. The fact that we have clauses of type (3) and (4) allows us to prove by induction that for every $k \in \omega$, there is exactly one σ of length k such that p_σ belongs to M. Moreover, an easy induction on k, l will show that the unique sequences σ, τ (of length k and l respectively) with $p_\sigma, p_\tau \in M$ are comparable. Thus, setting $\pi = \bigcup\{\sigma_k : p_{\sigma_k} \in M, \mid \sigma_k \mid = k\}$, we can easily check that $M = M_\pi$. This

shows that $\mathcal{S}(P) \subseteq \{M_\pi : \pi \in [T]\}$.

Finally, suppose that $\pi = \langle \pi_0, \pi_1, \ldots \rangle$ is an infinite path through T. It is easy to see that the presence of clauses of type (1) ensures that $M_\pi \subseteq F_{P,M}$. Moreover, it is not difficult to prove, by induction on the length of the M_π-deduction of the atom p that $p \in F_{P,M}$ implies $p \in M_\pi$. This shows that for every $\pi \in [T]$, $M_\pi \in \mathcal{S}(P)$. This completes the proof. □

As was the case with Thereom 3.1, the proof of Theorem 3.3 is uniform. Thus, implicit in the construction of Theorem 3.3, there is an algorithm computing from an index of a recursive tree $T \subseteq \omega^{<\omega}$, the index of a recursive program P with associated effective degree preserving one-to-one correspondence between $[T]$ and $\mathcal{S}(P)$. Moreover, we can modify this algorithm in much the same way that we modified the algorithm implicit in Theorem 3.1 to ensure that if e is not an index of a recursive tree, then our algorithm produces an index which is not the index of any recusive program. Thus, there is a one-to-one recursive function g such that $g(e)$ is an index of a recursive program P if and only if e is an index of a recursive tree $T \subseteq \omega^{<\omega}$. Moreover, if e is an index of a recursive tree T contained in $\omega^{<\omega}$ and P is the recursive program with index $g(e)$, then there is an associated effective one-to-one degree preserving correspondence between $[T]$ and $\mathcal{S}(P)$. Thus, the recursive function g shows that *Infpath* is 1-reducible to *Stab* and *Finpath* is 1-reducible to *Nostab*.

We thus have the following corollaries.

Corollary 3.4 *(a) Stab is 1-equivalent to Infpath.*

(b) Nostab *is 1-equivalent to Finpath.*

(c) Stab is a Σ^1_1*-complete set of natural numbers.*

(d) Nostab is a Π^1_1*-complete set of natural numbers.*

Proof: The functions f and g mentioned in the remarks following Theorem 3.1 and Theorem 3.3 respectively verify (a) and (b). It is proved in [Rogers, 1967] that $Finpath$ is a Π^1_1-complete set. It is easy to see, by writing out the definition, that $Nostab$ is a Π^1_1 set and hence $Nostab$ is a Π^1_1-complete set. Similarly, $Infpath$ is a Σ^1_1-complete set. Again one can write out the definition of $Stab$ to show that $Stab$ is Σ^1_1 set. So $Stab$ is a Σ^1_1-complete set. □

Corollary 3.5 *The set of all indices of recursive programs which possess at least two stable models is a Σ^1_1 set of natural numbers. Hence the collection of indices of programs possessing at most one stable model is a Π^1_1 set of natural numbers.*

Corollary 3.6 *The set of all indices of recursive programs which possess exactly one stable model is the intersection of a Π^1_1 set and a Σ^1_1 set, hence it is a Δ^1_2 set of natural numbers.*

The results of this section show that whenever we have a recursive program with the unique stable model, we can produce a recursive tree with a unique infinite branch such that the Turing degrees of the stable model and the branch are the same. Conversely, given a tree with a unique branch, we can produce a program with a unique stable model such that the Turing degrees of the branch and of the stable model are the same. Now if a recursive tree has a unique branch, the branch is hyperarithmetical. Hence, if a recursive program has a unique stable model, then that stable model is hyperarithmetical. This result can also be derived from the fact that $\mathcal{S}(P)$ is Π^0_2.

In recursion theory, recursive trees with a unique branch have been investigated previously by Clote, Cenzer and Smith, and Hinman. Here is a sample of the type of result that can be derived from their work. Hinman ([Hinman, 1978]) showed that for every re-

cursive ordinal α, there exist a tree $T_\alpha \subseteq \omega^{<\omega}$ such that T has a unique infinite path π and $\pi \equiv_T \mathbf{0}^{(\alpha)}$ (where $\mathbf{0}^{(\alpha)}$ is the α^{th} jump of $\mathbf{0}$ and $\mathbf{0}$ the degree of the recursive sets). So by Theorem 3.3 we get the following:

Corollary 3.7 *For each recursive ordinal α, there exists a recursive program P possessing a unique stable model M such that $M \equiv_T \mathbf{0}^{(\alpha)}$.*

It is a well known result of Kleene (see [Rogers, 1967], Theorem XLII(a)]) that every recursive tree $T \subseteq \omega^{<\omega}$, which has an infinite branch, has an infinite branch which is recursive in $Finpath$. However there is a recursive tree $T \subseteq \omega^{<\omega}$ such that $[T]$ is nonempty but $[T]$ has no hyperarithmetical elements. These facts plus Theorems 3.1 and Theorem 3.3 immediately imply the following.

Corollary 3.8 *(a) Every recursive program P which has a stable model has a stable model M such that $M \leq_T B$ where B is a complete Π^1_1-set.*
(b) There a recursive program P such that P has a stable model but P has no stable model which is hyperarithmetic.

4 Conclusion

We have characterized the set $\mathcal{S}(P)$ of stable models of a logic program by using effective descriptive set theory. This allows the latter subject to be applied to determine the complexity of the set of stable models of a logic program. Our results explain why the problem of testing whether a recursive program P possesses a stable model is complicated.

The fact that the class of stable models of a recursive program is a notational variant of sets in lower levels of arithmetical hierarchy in Baire space or Cantor space instead of ω is

new, and indicates that there are deep connections of "negation as failure" in nonmonotonic logic programming with classical effective descriptive set theory and therefore with recursion theory.

References

[Andreka and Nemeti, 1978] H. Andreka, I. Nemeti. The Generalized Completeness of Horn Predicate Logic as a Programming Language. *Acta Cybernetica* 4(1978) pp. 3-10.

[Apt, 1988] K.R. Apt. Introduction to Logic Programming. TR-87-35, University of Texas, 1988.

[Apt and Blair, 1990] K.R. Apt, H.A. Blair. Arithmetical Classification of Perfect Models of Stratified Programs. *Fundamenta Informaticae* 13(1990) pp. 1-17.

[Apt, Blair and Walker, 1987] K.R. Apt, H.A. Blair, A. Walker. Towards a theory of declarative knowledge. In: J. Minker ed. *Foundations of Deductive Databases and Logic Programming*, pp. 89-142, Morgan Kaufmann, Los Altos, CA.

[Bidoit and Froidevaux, 1988] N. Bidoit, C. Froidevaux. General logical databases and programs, default logic semantics, and stratification. *J. Information and Comput.*

[Clark, 1978] K.L. Clark. Negation as Failure. In: *Logic and Data Bases*, H. Gallier and J. Minker, eds. Plenum Press, New York, pp. 293–322.

[Doyle, 1979] J. Doyle. Truth Maintenance System. *Artificial Intelligence* 12(1979) pp. 231–272.

[van Emden and Kowalski, 1976] M.H. van Emden, R. Kowalski The Semantics of Predicate Logic as Programming Language. *Journal of Association for Computing Machinery* 23(1976) pp. 733–742.

[Gelfond and Lifschitz, 1988] M. Gelfond, V. Lifschitz. Stable Semantics for Logic Programs. In: *Proceedings of 5th International Symposium Conference on Logic Programming*, Seattle, 1988.

[Hinman, 1978] P. Hinman. *Recursion-theoretic Hierarchies*. Springer Verlag, 1978.

[Marek and Nerode, 1990] W. Marek, A. Nerode. Decision procedure for default logic Mathematical Sciences Institute Report, Cornell University.

[Marek, Nerode and Remmel, 1990] W. Marek, A. Nerode, and J. Remmel. A theory of nonmonotonic rule systems I. *Annals of Mathematics and Artificial Intelligence* 1(1990) pp. 241-273.

[Marek, Nerode and Remmel, 1990a] W. Marek, A. Nerode, and J. Remmel. A theory of nonmonotonic rule systems II. TR 9-32, Mathematical Sciences Institute at Cornell University (to appear in *Annals of Mathematics and Artificial Intelligence (1991)*).

[Marek, Nerode and Remmel, 1991] W. Marek, A. Nerode, and J. Remmel. Classyfying rule systems according to the number of derivations of elements. In preparation.

[Marek and Subrahmanian, 1988] W. Marek, V.S. Subrahmanian. The Relationship Between Logic Program Semantics and Non-Monotonic Reasoning. To appear in: *Theoretical Computer Science*.

[Marek and Truszczyński, 1989] W. Marek and M. Truszczyński. Stable semantics for logic programs and default theories Proceedings of the North American Conference on Logic Programming pp. 243–256 MIT Press, 1989.

[Minker, 1987] J. Minker. *Foundations of Deductive Databases and Logic Programming* Morgan Kaufmann, Los Altos, CA, 1987.

[Reiter, 1980] R. Reiter. A Logic for Default Reasoning. *Artificial Intelligence*, 13(1980) pp. 81-132.

[Rogers, 1967] H. Rogers, Jr. *Theory of Recursive Functions and Effective Computability* Mc Graw-Hill, 1967.

[Smullyan, 1961] R.M. Smullyan. *Theory of Formal Systems*, *Annals of Mathematics Studies*, no. 47, Princeton, N.J.